heavily loaded and for improved responsiveness in the case where the computing system is not heavily loaded.

[0010] In accordance with one embodiment of the invention and with reference to FIG. 1, a process generally designated 100 for executing operations upon a data structure includes a first step 110 in which operations upon a data structure are received and queued in a task queue. Such operations include insertion, deletion and update operations which can further be divided into distinct operation tasks as is well known in the art. Update operations, which include modification of a value of a data structure element, can be executed as an insertion of a new data structure element followed by deletion of an old data structure element.

[0011] In a step 120 the process 100 checks if any operations have been received in step 110. If any operations have been received, then in a step 130 the update phase state transitions of the received operations are either executed or postponed in the update phase as further described herein. Operations postponed are returned to the task queue. In a step 140 the commit phase state transitions of the received operations which have not been postponed are executed in the commit phase. If no operations have been received as determined in step 120, the process 100 returns to step 110 and waits to receive operations.

Upon completion of the commit phase in step 140, the check performed in step 120 is repeated.

[0012] During the execution of the update phase of step 130 and the commit phase of step 140, additional operations may be received and queued by step 110 for later execution in steps 130 and 140. Thus the number of operations executed in steps 130 and 140 is dependent upon the number of operations queued in step 110. Process 100 thus executes operations in such manner that if a small number of operations are queued in step 110, a small number of operations are executed in steps 130 and 140 and if a large number of operations are queued in step 110, a large number of operations are executed in steps 130 and 140.

[0013] It has been found that the process 100 operates to improve the efficiency of the computing system. In the case where relatively few operations are received in step 110, then the process 100 achieves improved responsiveness. If on the other hand, a relatively large amount of operations are received in step 110, then the process 100 achieves improved efficiency by reducing the number of times the process 100 switches between the update phase of step 130 and the commit phase of step 140.

[0014] The invention may be practiced in a computing system 200 such as illustrated in FIG. 2. Computing system 200 includes computer code 220 having computer instructions for performing processes of the invention, and memory 230 for storing task

queue 260, update data 250, and a data structure 240, as further described herein. Computer code 220 is optionally stored in memory 230. Computing system 200 and the components included therein are optionally distributed over a plurality of computing devices.

[0015] Data structure 240 is preferably a linked structure and includes a plurality of elements 245, such as nodes or list elements as is well known in the art. Elements 245 further include fields 247 such as data fields and links which enable the linked structure. While data structure 240 has been described in terms of nodes and list elements, those skilled in the art will recognize that data structure 240 may include any data organization that uses links or the like to associate and establish relationships between data structure elements 245.

[0016] According to one embodiment of the invention, operations upon the data structure 240 include state transitions between defined states as listed in Table 1. An element 245 of data structure 240 affected by the execution of a given operation includes a corresponding state field preferably stored in field 247.